Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Marek Futrega
Nr albumu: 174715

Internetowy serwis gier sieciowych

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr Janiny Mincer-Daszkiewicz
Instytut Informatyki

czerwiec 2003

Pracę przedkładam do oceny

Data Podpis autora pracy:

Praca jest gotowa do oceny przez recenzenta

Data Podpis kierującego pracą:

Streszczenie

Tematem pracy jest internetowy serwis gier sieciowych, który umożliwia internautom rozgrywanie w czasie rzeczywistym partii w gry planszowe i karciane. Implementacja serwisu opisanego w pracy trwała od czerwca 2001 r., a jej efekty są dostępne na stronie `www.kurnik.pl`. Od strony serwera na implementację składają się serwery gier o rozproszonej architekturze napisane w języku C++, baza danych oparta o serwer MySQL, pośrednik bazodanowy w Javie oraz skrypty w PHP do generowania dynamicznych stron WWW; od strona klienta implementacja obejmuje aplety w Javie.

Słowa kluczowe

Internet, gry, Java, PHP, MySQL

Klasyfikacja tematyczna

H. Information Systems
H.4. INFORMATION SYSTEMS APPLICATIONS
H.4.3. Communications Applications

Spis treści

Wstęp

Gry planszowe towarzyszą ludziom od około 4 tysięcy lat i są jednym z popularnych sposobów spędzania wolnego czasu (zob. [GTG]). Z chwilą, kiedy Internet stał się powszechnie dostępny, pojawiała się możliwość grania w czasie rzeczywistym na odległość. Wcześniej nie lada problemem mogło być znalezienie w okolicy chętnego na partyjkę w szachy czy warcaby, nie mówiąc już o kimś na równym lub wyższym poziomie; z pojawieniem się gry na odległość problem przestał istnieć — przeciwnikiem mógł być ktoś choćby i po drugiej stronie kuli ziemskiej, byleby był użytkownikiem Internetu.

Niniejsza praca przedstawia projekt i implementację serwisu internetowego, który umożliwia grę przez Internet w czasie rzeczywistym w gry planszowe i karciane (ogólnie turowe). Przez serwis internetowy należy tutaj rozumieć zarówno strony WWW serwisu, jak również oprogramowanie do rozgrywania gier (klienckie i serwerowe), bazę danych itp.

W rozdziale 1. przedstawione zostały pokrótce wymagania serwisu (funkcjonalne, jak i niefunkcjonalne). Rozdział 2. zawiera ogólny projekt serwisu; opisana została architektura systemu, organizacja gier i obsługa użytkowników w serwisie, struktura serwisu WWW i bazy danych. Opis implementacji serwisu, ze zwróceniem uwagi na ciekawsze jej aspekty, przedstawiony został w rozdziale 3. W kolejnym rozdziale umieszczone zostały informacje związane z wdrożeniem serwisu (historia jego rozwoju, statystyki itp.). Rozdział 5. to podsumowanie pracy, w szczególności ocena realizacji wymagań.

W tym miejscu chciałbym złożyć podziękowania mojemu opiekunowi naukowemu, dr Janinie Mincer-Daszkiewicz, za pomoc w powstaniu tej pracy, w szczególności za cenne uwagi do kolejnych wersji pracy.

Rozdział 1

Wymagania

1.1. Wymagania funkcjonalne

Pierwszym i podstawowym wymogiem serwisu z grami sieciowymi jest oczywiście danie użytkownikom Internetu możliwości rozgrywania między sobą gier w czasie rzeczywistym. Poza uczestnictwem w grze, użytkownik powinien również mieć możliwość oglądania gier rozgrywanych przez innych użytkowników serwisu (jest to bardzo pomocne w nauce gry), o ile nie będą oni mieli nic przeciwko temu.

Prostota i wygoda korzystania z serwisu to wymogi przesądzające o tym, jak duże będzie grono jego użytkowników. Jeśli przed pierwszym przystąpieniem do gry użytkownik będzie musiał pobrać program do grania (w wersji źródłowej), skompilować go, zainstalować, skonfigurować, a następnie uruchomić, to oczywiście liczba potencjalnych użytkowników serwisu będzie istotnie ograniczona. Idealnie byłoby, gdyby użytkownik nie musiał pobierać wcześniej żadnego oprogramowania, tzn. gdyby gra odbywała się na stronie WWW. Takie rozwiązanie nie jest jednak możliwe, ponieważ strony mają z natury charakter statyczny — treści na nich zawarte nie zmieniają się do czasu pobrania nowej wersji z serwera WWW. Z kolei odświeżanie co jakiś czas strony z aktualnym stanem gry posiada poważną wadę — za każdym razem przesyłany jest cały aktualny stan gry zamiast jedynie zmian stanu, które w grach są zazwyczaj nieznaczne (np. przestawienie piona z pozycji X na Y). Konieczne jest zatem zastosowanie dedykowanego oprogramowania po stronie klienta. Proces pobierania, instalacji i uruchamiania takiego oprogramowania może być bardzo uproszczony, jeśli osadzi się je na stronie WWW — od użytkownika wymagane będzie wtedy jedynie wejście na stronę, a zatem niewiele.

Równie małe powinny być wymagania związane z interakcją. Użytkownik zamiast wpisywać magiczne komendy, które wcześniej musiałby sobie przyswajać, powinien prowadzić grę np. przestawiając interesujące figury na ekranie za pomocą myszki. Nie obędzie się zatem bez odpowiedniej wizualizacji stanu gry; należy jednak przy tym uważać z ogólną atrakcyjnością wizualną. W przypadku typowych gier komputerowych w sporym stopniu decyduje ona o tym, czy gra się spodoba i czy użytkownik do niej jeszcze powróci; w przypadku gier umysłowych, jakimi są gry planszowe i karciane, nie ma ona większego znaczenia, a nawet wręcz przeciwnie — bajecznie kolorowa, animowana grafika wokół planszy szachowej może skutecznie zniechęcić niejednego amatora szachów. Co więcej, w przypadku gier internetowych osadzonych na stronach im mniejsza liczba "wodotrysków", tym mniejszy ich rozmiar i, co za tym idzie, krótszy czas ładowania (uruchamiania).

Serwis z grami powinien również zapewniać użytkownikom możliwość stałej identyfikacji. Dzięki temu za każdym razem, kiedy ktoś natrafi na "godnego siebie przeciwnika", będzie

mógł łatwo go rozpoznać następnym razem. Użytkownik będzie zatem musiał rejestrować się w serwisie, wybierając sobie unikatowy identyfikator; powinna jednak istnieć także opcja zagrania anonimowo, bez konieczności wcześniejszej rejestracji (celem wypróbowania serwisu).

Każdy użytkownik powinien sam decydować o tym, z kim będzie grał; jedni bowiem lubią grać z równymi, inni z dużo lepszymi od siebie, a jeszcze inni — z gorszymi. Aby móc ocenić, kto jest lepszy, a kto gorszy, potrzebny będzie system oceny siły graczy. Najlepiej do tego nadaje się chyba ranking wzorowany na rankingu szachowym (zob. [ELO]), w którym gracze otrzymują na początku określoną liczbę punktów, a następnie z każdą rozegraną partią liczba ta rośnie bądź maleje w zależności od wyniku partii i siły przeciwnika. Poza rankingiem przydatne mogą być też inne informacje o użytkowniku, jak liczba dotychczasowych zwycięstw i porażek, historia rozegranych gier, ich przebiegi, lista przeciwników z bilansem gier z każdym z nich itp.

Kolejnym wymogiem funkcjonalnym jest możliwość tekstowej komunikacji (tj. przesyłania krótkich wiadomości) pomiędzy użytkownikami korzystającymi w danej chwili z gier. Obecna powinna być zarówno komunikacja typu jeden do wielu (publiczna), jak również jeden do jeden (prywatna).

1.2. Wymagania niefunkcjonalne

Dostęp do treści publikowanych na stronach WWW powinien być z założenia niezależny od używanego systemu operacyjnego, przeglądarki, architektury procesora itp. Oprogramowanie klienckie do gier osadzone na stronach serwisu powinno też spełniać to założenie, nawet mimo tego, że ponad 90 procent użytkowników Internetu korzysta z jednakowego systemu i przeglądarki (zob. [RNK]).

Po stronie serwera niezależność oprogramowania od architektury, systemu itp. nie jest aż tak istotna, choć też jest mile widziana (przynajmniej na poziomie kodu źródłowego), bowiem z czasem może pojawić się konieczność migracji serwisu, w całości lub w części, na inną platformę sprzętową bądź systemową.

Czas reakcji na akcje podejmowane przez użytkownika (np. wykonanie ruchu w grze) powinien być jak najkrótszy — najlepiej tak krótki, aby nie były zauważalne opóźnienia od podjęcia akcji do jej wykonania. Podobnież krótki powinien być czas otwierania stron WWW serwisu (tych z dynamicznie generowaną zawartością, np. statystykami), ponieważ jest to jeden z czynników stanowiących o komforcie korzystania z serwisu.

Z kolei czynnikiem stanowiącym o atrakcyjności serwisu z grami jest ich liczba i dobór. Oprogramowanie powinno być skonstruowane w taki sposób, aby można było prosto i szybko dodawać nowe gry przez rozszerzanie pewnych ogólnych szkieletów gier (generycznych klas).

Dwa inne wymagania niefunkcjonalne serwisu to: wydajność (aby na danej platformie sprzętowej móc obsługiwać jak największą liczbę jednoczesnych użytkowników) oraz skalowalność (aby nie istniało programowe ograniczenie liczby użytkowników jednocześnie korzystających z serwisu; ciągły wzrost tej liczby wymagałby jedynie rozbudowy platformy sprzętowej).

Gotowe oprogramowanie wykorzystywane do budowy serwisu (serwer bazy danych, serwer WWW itp.) powinno być maksymalnie dostosowywalne do własnych potrzeb (najlepiej z ogólnie dostępnymi kodami źródłowymi, prawem ich modyfikacji itp.). Nie bez znaczenia jest również koszt licencji takiego oprogramowania.

Rozdział 2

Projekt

W niniejszym rozdziale przedstawiono projekt serwisu gier sieciowych, który spełnia wymagania określone w poprzednim rozdziale.

2.1. Ogólna architektura systemu

W przypadku typowych serwisów internetowych mamy do czynienia z architekturą klient-serwer, gdzie rolę klienta pełni **przeglądarka WWW**, a serwera — **serwer WWW**. Przeglądarka łączy się z serwerem, pobiera strony wskazane przez użytkownika, po czym je prezentuje.

Program, w którym przebiega właściwa gra, tzw. **klient gry**, osadzony jest na stronie WWW. Kiedy użytkownik korzysta z gier, klient utrzymuje stałe połączenie z **serwerem gry**, którego zadaniem jest koordynacja rozgrywanych gier i połączonych z nim użytkowników. Dla każdej z gier (szachy, warcaby itp.) istnieje indywidualny serwer gry, dzięki czemu problemy ze stabilnością jednego z nich nie mają wpływu na działanie pozostałych.

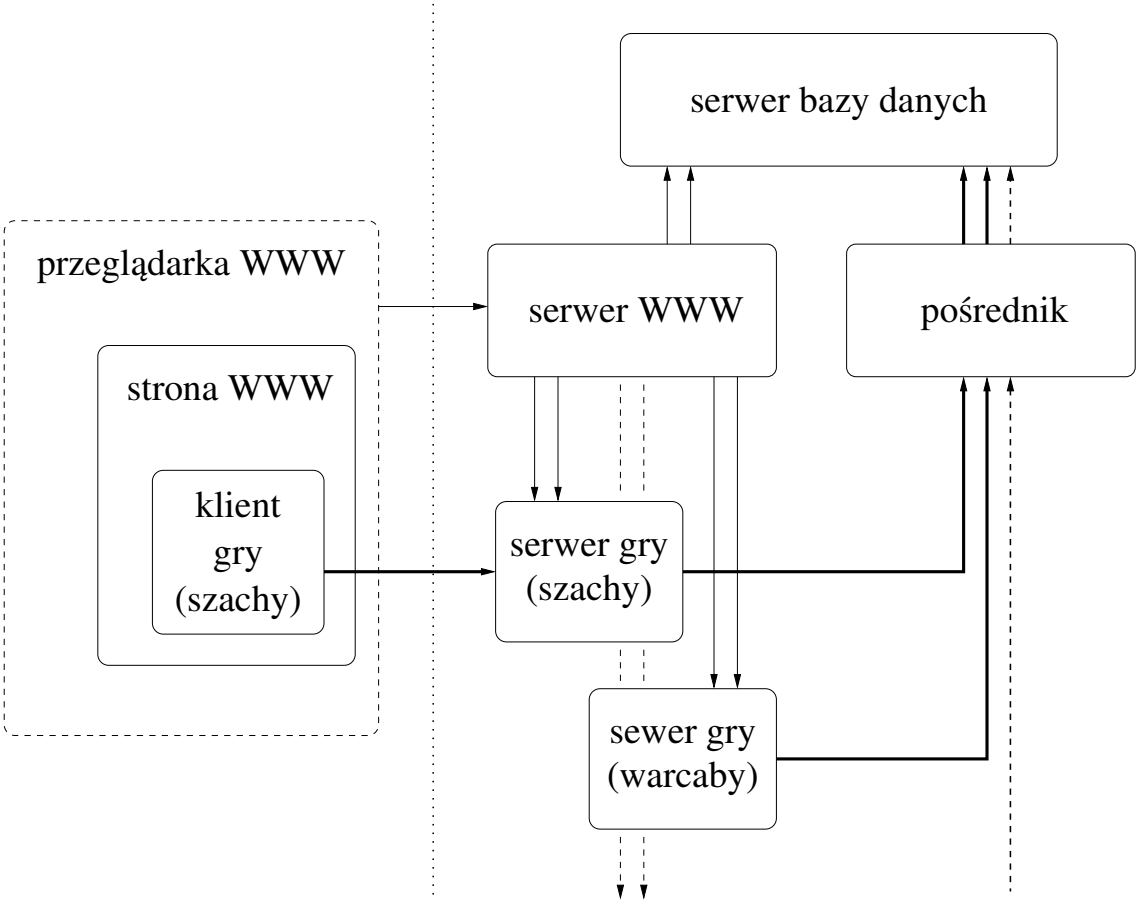
Dane o użytkownikach, rozegranych partiach itp. przechowuje **serwer bazy danych**, do którego dostęp posiada serwer WWW oraz serwery gier. W przypadku serwerów gier dostęp ten nie jest jednak bezpośredni — wszelkie operacje bazodanowe wykonuje za nie **pośrednik**.

Ogólny schemat systemu przedstawiono na rys. 2.1; stałe połączenia między elementami systemu wyróżnione są przez pogrubienie; pomiędzy pewnymi elementami może istnieć więcej niż jedno połączenia — na rysunku jest to zobrazowane przez obecność dwóch strzałek.

2.1.1. Serwer gry

Zadaniem serwera gry jest koordynacja gier rozgrywanych przez użytkowników serwisu. Z serwerem kontaktuje się każdy klient gry i przekazuje mu informacje o działaniach podejmowanych przez użytkownika; otrzymując informacje od klientów, serwer wykonuje określone operacje i o ewentualnych zmianach stanu (np. stanu rozgrywanej partii) informuje klientów.

Żądania, jakie przychodzą od klientów gry, muszą być przez serwer obsługiwane w jak najkrótszym czasie; w przeciwnym razie opóźnienia w działaniu mogłyby być zauważalne przez użytkowników i wpływałoby to negatywnie na komfort gry. Aby uniknąć przestojów serwera związanych z operacjami bazodanowymi, wszelkie takie operacje są zlecane pośrednikowi, którego zadaniem jest komunikacja z bazą danych. Serwer otrzymuje tylko wyniki tych operacji, kiedy są już gotowe; w czasie ich wykonywania może kontynuować obsługiwanie żądań od użytkowników.

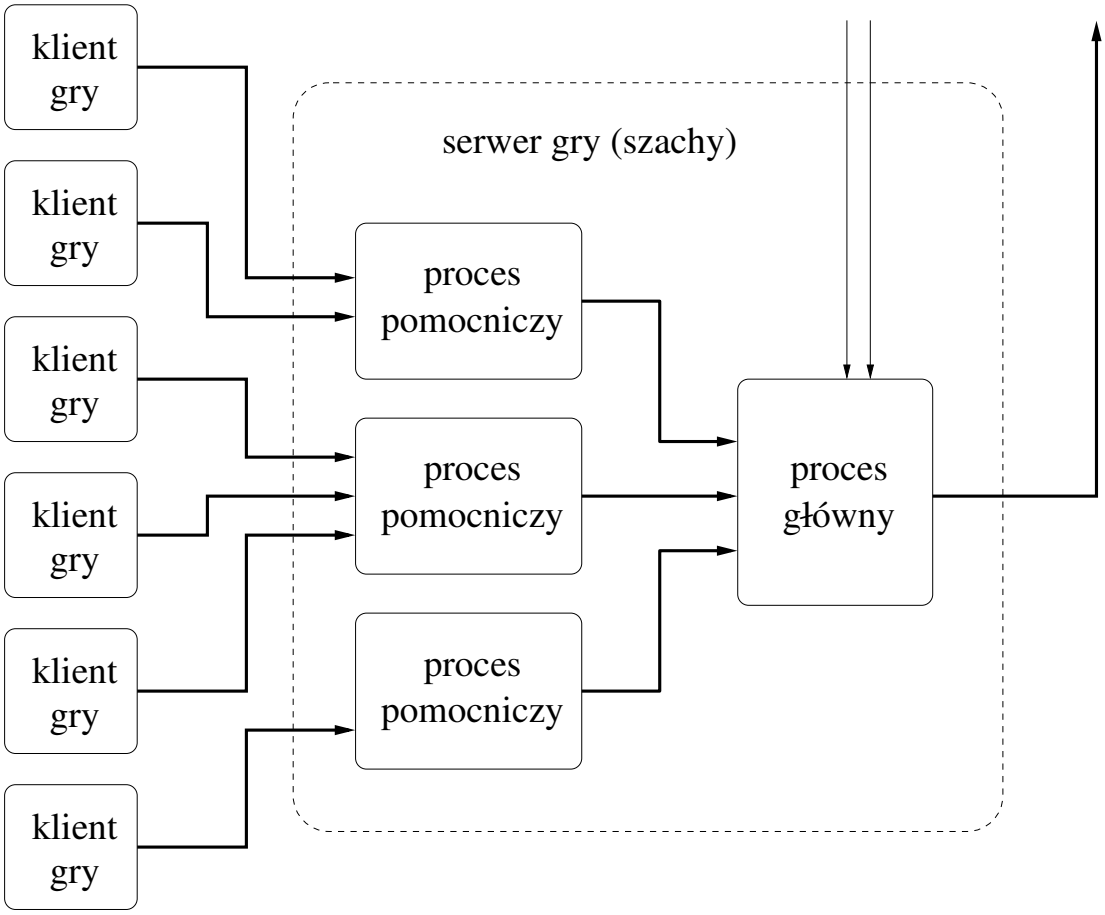


Rysunek 2.1: Ogólna architektura systemu

Serwer gry ma architekturę rozproszoną i składają się nań (dokładnie) jeden proces główny i co najmniej jeden proces pomocniczy. Schemat serwera gry przedstawiono na rys. 2.2.

Proces główny utrzymuje stan rozgrywanych partii, listę użytkowników z nim połączonych itp. Dzięki temu, że istnieje dokładnie jeden taki proces, nie występują problemy z synchronizacją dostępu do danych, propagacją danych i inne wynikające z rozproszenia danych.

Zadaniem procesów pomocniczych jest przede wszystkim utrzymywanie bezpośrednich połączeń z klientami gier i przekazywanie otrzymywanych od nich żądań do procesu głównego. Ich istnienie może się wydawać z pozoru zbyteczne, ale w rzeczywistości jest konieczne ze względu na istnienie systemowych ograniczeń (np. ograniczenie na liczbę jednoczesnych połączeń, jakie może utrzymywać jeden proces). Każdy proces pomocniczy może obsługiwać wielu jednoczesnych klientów, a do przekazywania informacji od nich do procesu głównego potrzebuje tylko jednego połączenia. Dzięki temu, że mamy tutaj do czynienia z procesami, a nie wątkami, istnieje możliwość ich rozpraszania na różne serwery fizyczne. To z kolei oznacza, że praktycznie jedyne ograniczenie na liczbę jednoczesnych użytkowników wynika z ograniczeń serwera fizycznego, na którym działa proces główny serwera gier. Obecność procesów pomocniczych ma również dodatkową zaletę — proces główny może bowiem obsługiwać w sposób przezroczysty wszystkich klientów niezależnie od tego, w jaki sposób komunikują się oni z procesami pomocniczymi (bezpośrednie połączenia TCP/IP, tunelowanie przez HTTP itp.)



Rysunek 2.2: Schemat serwera gry

2.1.2. Pośrednik

Obecność w systemie pośrednika odpowiadającego za komunikację z bazą danych wynika z chęci zwolnienia serwerów gier z konieczności wykonywania przez nie operacji bazodanowych, które uniemożliwiałyby im natychmiastowe obsługiwanie żądań od klientów. Praca pośrednika sprowadza się do otrzymywania zleceń, wykonywania odpowiednich operacji na bazie danych i odsyłania rezultatów, jeśli takowe są wymagane.

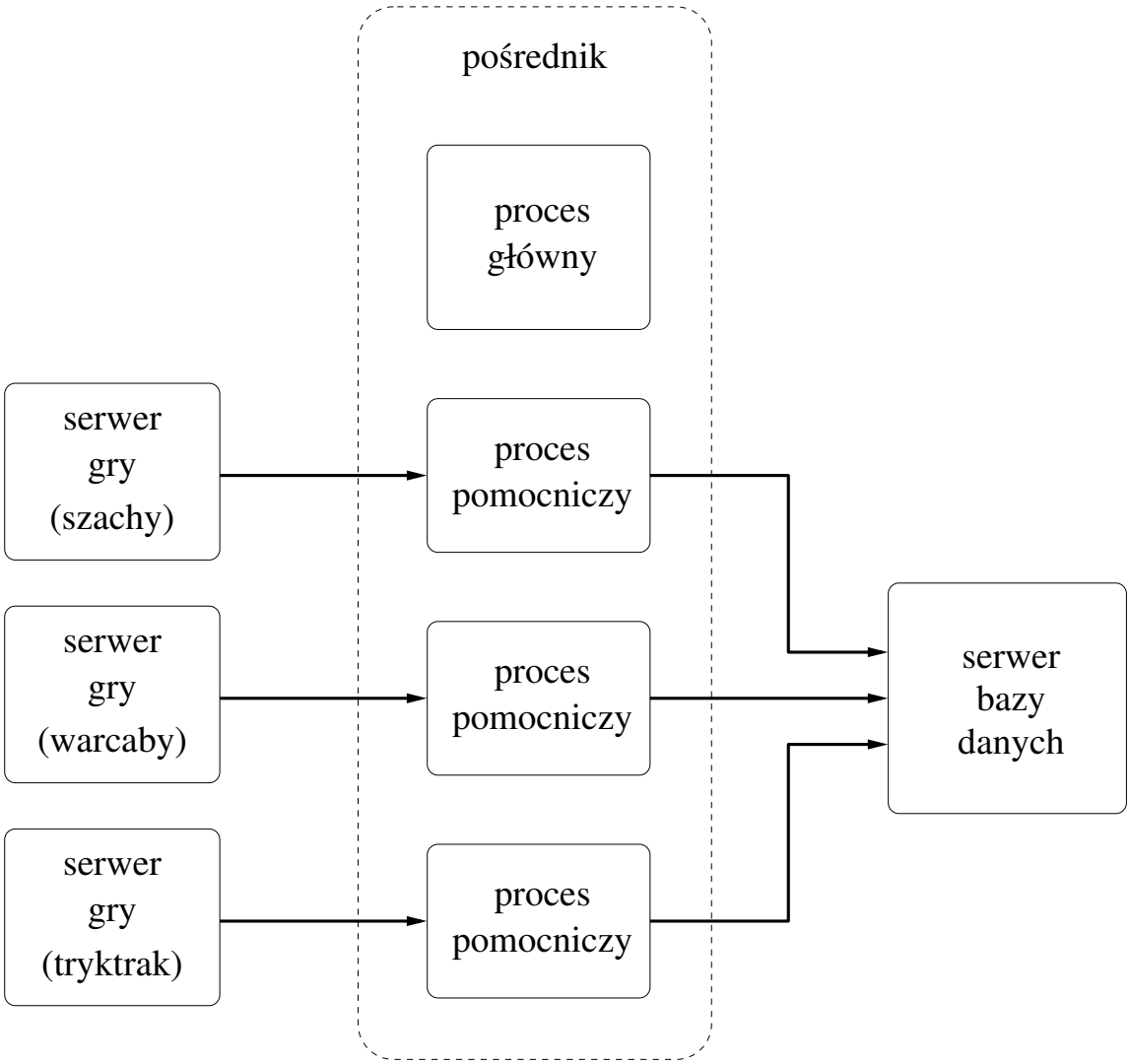
Pośrednik zorganizowany jest w ten sposób, że dla każdego z serwerów gier istnieje jeden proces zajmujący się realizacją zleceń. Dzięki temu, że nie jest to jeden proces, który realizuje zlecenia od wszystkich serwerów gier, większość operacji bazodanowych może być wykonywana równolegle, o ile pozwala na to sam serwer bazy danych W skład pośrednika wchodzi również proces główny, którego rola sprowadza się do akceptacji nowych połączeń od serwerów gier i nadzoru nad procesami pomocniczymi (realizującymi zlecenia).

Schemat pośrednika przedstawiono na rys. 2.3.

2.1.3. Klient gry

Program osadzony na stronie, przez który odbywa się właściwa gra, został zaprojektowany jako tzw. "cienki klient". Jego zadania to komunikacja z serwerem, prezentacja stanu gry i obsługa zdarzeń od użytkownika.

W kliencie nie zawarto samej logiki gry — znajduje się ona w całości po stronie serwera gry,



Rysunek 2.3: Schemat pośrednika

co jest niewątpliwą zaletą (nie ma możliwości powstawania konfliktów; łatwiej wprowadzać ewentualne zmiany). Ponadto mniejszy jest dzięki temu rozmiar programu, a to z kolei oznacza krótszy czas jego ładowania i uruchamiania przez Internet.

Część klienta gry jest wykorzystywana w serwisie także do odtwarzania gier już rozegranych.

2.2. Organizacja gier i obsługa użytkowników na serwerze gry

Po połączeniu z serwerem gry, użytkownik widzi następujące elementy: listę użytkowników połączonych w danej chwili z serwerem gry; listę "wirtualnych stołów", przy których rozgrywane są gry; publiczne wypowiedzi użytkowników.

Na liście użytkowników poza identyfikatorami graczy znajduje się też informacja o ich aktualnym rankingu i numerze stołu, przy którym się znajdują. Lista może być sortowana według dowolnego z kryteriów (identyfikator, ranking, numer stołu).

Powodem istnienia abstrakcji stołów jest wprowadzenie organizacji gier, która naśladuje

sposób organizacji gier w rzeczywistości. Każdy użytkownik może przyłączyć się do wybranego przez siebie stołu; może również założyć własny stół, jeśli chce. Użytkownik obecny przy stole może zająć przy nim miejsce, jeśli jest wolne i ma chęć zagrać. Kiedy komplet graczy zajmie miejsca przy stole (zazwyczaj dwóch), mogą oni rozpocząć grę za obopólną zgodą. Użytkownicy obecni przy stole, którzy nie zajmują przy nim miejsca, mogą jedynie przyglądać się rozgrywanej partii. Obserwatorzy mogą przychodzić do stołu i od niego odchodzić w dowolnym momencie gry; uczestnicy jednak nie mogą jej opuszczać do czasu jej zakończenia.

Gracz, który zakłada stół, zostaje jego operatorem. Daje mu to prawo ustawiania parametrów stołu oraz gry, która będzie miała przy nim miejsce. Istnieją trzy rodzaje stołów: *publiczny*, *prywatny* i *chroniony*. Pierwszy z nich to taki, przy którym każdy może przebywać i każdy może zajmować miejsce; przy drugim również może przebywać każdy, ale miejsce zajmować mogą tylko użytkownicy zaproszeni przez operatora stołu; przy trzecim zaproszenie operatora wymagane jest zarówno do zajęcia miejsca, jak i do samej obecności przy stole. Operator może zapraszać do stołu wybrane osoby przez wskazanie ich na liście użytkowników; wybrana osoba otrzymuje zaproszenie, które może zaakceptować bądź odrzucić; jeśli je zaakceptuje, dołącza do stołu. Operator może również wypraszać wybrane osoby ze stołu, nie może jednak wyprosić żadnej z osób, która uczestniczy w grze.

Istnieją dwa rodzaje gier, jakie mogą być rozgrywane przy stole: gry *rankingowe* oraz *towarzyskie*. W przypadku gier rankingowych, wynik partii powoduje zmiany rankingu graczy, a informacja o jej rozegraniu i sam przebieg odnotowywane są w statystykach obu graczy. Niekiedy partie rankingowe powodują też zmiany związane z dodatkowym systemem mierzenia siły graczy, tzw. drabinką. Partie towarzyskie nie mają wpływu na ranking, nie są też odnotowywane żadne informacje o ich odbyciu.

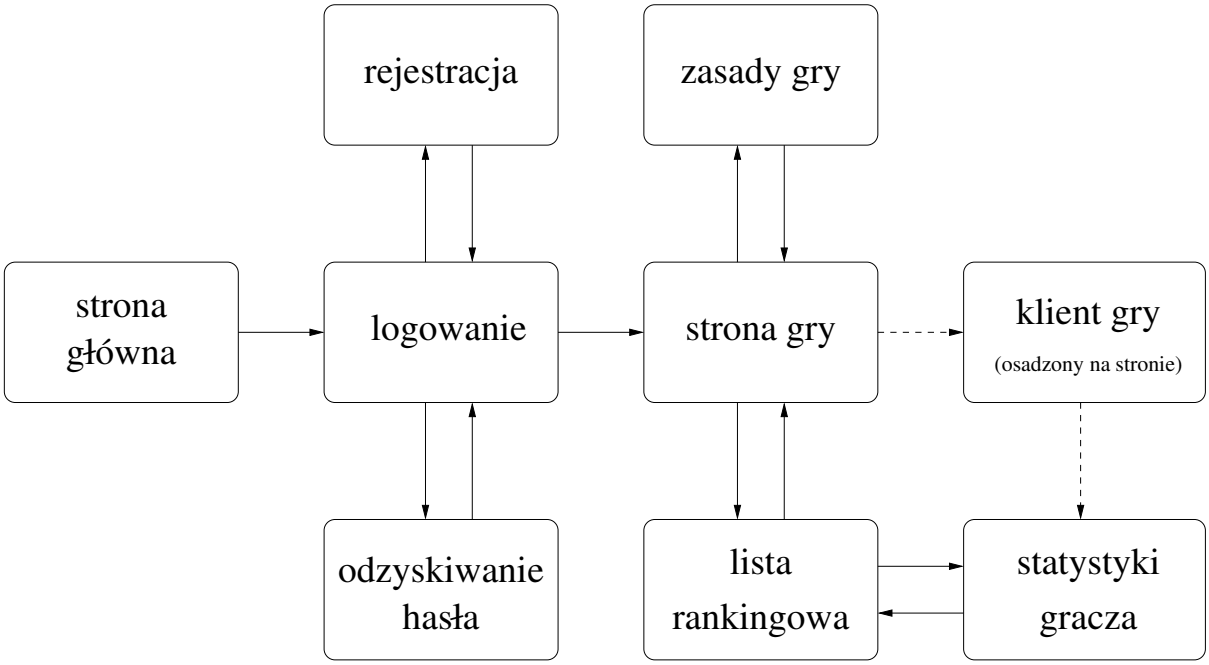
Istnieją trzy rodzaje komunikacji, jaką mogą prowadzić użytkownicy połączeni z serwerem gry:

1. komunikacja jeden do wielu (publiczna), gdzie wiadomość tekstowa użytkownika jest przesyłana do wszystkich innych użytkowników znajdujących się na liście; wypowiedzi publiczne prezentowane są w kolejności ich wysłania (najnowsze na dole) i każda z nich jest poprzedzona identyfikatorem osoby, która ją wysłała;

2. komunikacja jeden do wielu ograniczona do osób obecnych przy tym samym stole do gry;

3. komunikacja jeden do jeden — użytkownik może prowadzić w ten sposób prywatną rozmowę z dowolnym innym użytkownikiem obecnym w danym momencie na serwerze.

Po zakończeniu każdej gry typu rankingowego ulega zmianie ranking graczy, którzy brali w niej udział. Ponieważ lista użytkowników poza identyfikatorami osób przedstawia także ich aktualny ranking, zmiany te muszą być za każdym razem przesyłane do każdego z użytkowników. I tu pojawia się pewne niebezpieczeństwo. Jeśli bowiem średni czas trwania jednej partii będzie wynosił np. 5 min., jednocześnie z serwerem gry będzie połączonych 6 tys. użytkowników, z których każdy będzie rozgrywał kolejne partie, to 3 tys. partii na 5 min. daje 10 partii na sekundę, a to z kolei wymaga rozesłania 10 zmian rankingu na sekundę do wszystkich 6 tys. użytkowników! Takie działanie generowałoby spory ruch, który mógłby nawet uniemożliwiać grę osobom łączącym się przez modemy. Z tego powodu wprowadza się abstrakcję wirtualnych pokoi, które mają ograniczoną pojemność (100-200 osób). Przed połączeniem z serwerem gry, użytkownik wybiera pokój, w którym chce przebywać, a w pokoju zamiast listy wszystkich użytkowników połączonych z serwerem gry, widzi jedynie tych z pokoju. Tylko z nimi może się komunikować i grać. W ten prosty sposób ilość informacji, jakie musiałby otrzymywać (o zmianach rankingu, publicznych wypowiedziach itp.), jest kilkukrotnie mniejsza.



Rysunek 2.4: Struktura serwisu WWW

2.3. Struktura serwisu WWW

Strony serwisu WWW są zorganizowane w sposób przedstawiony na rys. 2.4. Strzałki symbolizują nawigację w serwisie; te z nich, które są narysowane przerywaną linią, oznaczają, że strony otwierane są w nowych oknach przeglądarki (domyślnie nowa strona zastępuje poprzednią).

Strona główna zawiera przede wszystkim listę dostępnych gier — użytkownik wybiera jedną z nich, a następnie loguje się w serwisie, podając swój identyfikator i hasło. Jeśli korzysta z serwisu po raz pierwszy, to musi się wcześniej zarejestrować. Rejestracja polega na wybraniu własnego identyfikatora oraz hasła i opcjonalnym podaniu daty urodzenia z adresem e-mail (wykorzystywane do odzyskiwania zapomnianych haseł); wybrany identyfikator musi być unikatowy, tzn. niezarejestrowany wcześniej. Po udanej rejestracji użytkownik powraca do strony logowania, a po zalogowaniu przechodzi na stronę wybranej gry.

Na stronie gry następuje wybór pokoju, w którym użytkownik chciałby zagrać. Wybranie pokoju powoduje otworzenie nowego okna przeglądarki, w którym osadzony jest klient gry. Poza wyborem pokoju, użytkownik może też przejść na stronę z zasadami gry bądź listą rankingową.

Strona z listą rankingową jest generowana dynamicznie na podstawie aktualnych danych pobieranych bezpośrednio z serwera bazy danych. Na liście poza identyfikatorem gracza i aktualnym rankingiem znajdują się również inne informacje, jak liczba zwycięstw, porażek, remisów; wskaźnik passa; czas ostatniej wizyty itp. Z listy rankingowej użytkownik może przejść na stronę ze statystykami wybranej osoby; można też na nią przejść bezpośrednio z klienta gry.

Strona ze statystykami użytkownika przedstawia jeden z trzech widoków (istnieje możliwość przełączania się między nimi) zawierających:

1. ogólne informacje o użytkowniku (aktualny ranking, liczba rozegranych gier, czas ostatniej wizyty itp.)

2. listę ostatnio rozegranych partii; dla każdej partii podana jest data i czas trwania, uczestnicy oraz wynik; w grach planszowych istnieje możliwość obejrzenia przebiegu wybranej partii; przebieg partii przedstawiany jest w nowym oknie przeglądarki, w którym osadzona jest część klienta gry; listę rozegranych partii można ograniczyć do pokazywania tylko zwycięstw, porażek bądź remisów;
3. listę przeciwników i (w grach partnerskich) listę partnerów; na liście znajdują się: identyfikator przeciwnika; jego aktualny ranking; liczba zwycięstw, porażek, remisów wybranego użytkownika z danym przeciwnikiem; ogólna liczba zwycięstw, porażek i remisów przeciwnika; czas od ostatniej wizyty.

2.4. Struktura bazy danych

W bazie danych przechowywane są informacje o zarejestrowanych użytkownikach oraz rozegranych partiach. Oto lista wykorzystywanych do tego tabel z krótką informacją o ich zawartości.

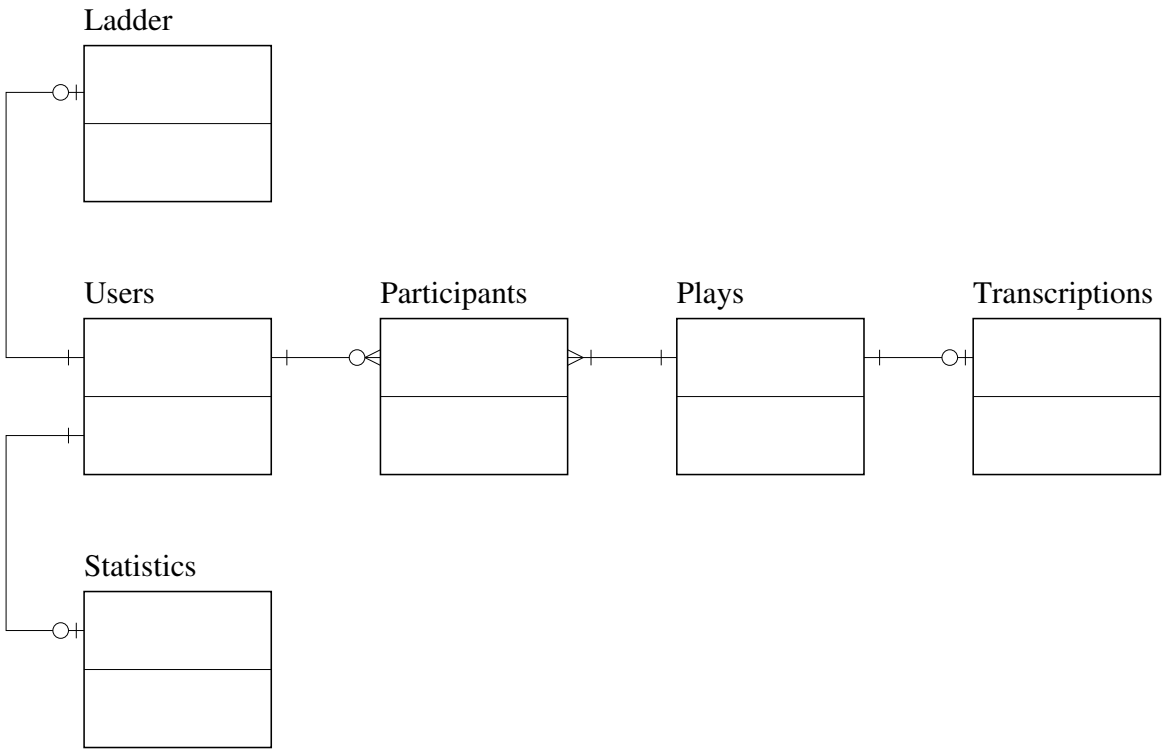
- tabela `Users` — zarejestrowani użytkownicy,
- tabela `Statistics` — statystyki użytkowników (liczba rozegranych gier, ranking itp.),
- tabela `Plays` — informacje o rozegranych partiach (data, wynik itp.),
- tabela `Participants` — uczestnicy partii,
- tabela `Transcriptions` — przebiegi partii,
- tabela `Ladder` — informacje o kolejności graczy na drabince,
- tabela `LadderTimes` — daty ostatnich kasowań drabinek,
- tabela `Cookies` — identyfikatory sesji zalogowanych użytkowników,
- tabela `LastVisits` — daty ostatnich wizyt użytkowników.

Relacje między wybranymi tabelami przedstawiono na diagramie RDR (ang. *Data Relationship Diagram*) na rys. 2.5.

Format przechowywania przebiegów partii w tabeli `Transcriptions` zależny jest od rodzaju gry. Dzięki zastosowaniu formatów standardowych dla poszczególnych gier użytkownicy serwisu mają możliwość pobierania plików z przebiegami i analizowania ich w zewnętrznych programach do analizy partii. Wykorzystywane formaty to:

- `PGN` (*Portable Game Notation*) dla gry szachy,
- `PDN` (*Portable Draughts Notation*) dla gry warcaby,
- `PBN` (*Portable Bridge Notation*) dla gry brydż,
- `MAT` dla gry tryktrak,
- `SGF` dla gry go.

Dla gier, dla których nie istnieją standardowe formaty zapisu przebiegu partii (np. młynek), używany jest format zbliżony do `PGN`.



Rysunek 2.5: Schemat części bazy danych

Rozdział 3

Implementacja

W niniejszym rozdziale przedstawiono implementację serwisu zgodną z projektem z poprzedniego rozdziału. Pierwsza wersja serwisu została uruchomiona w czerwcu 2001 r. (historię jego rozwoju dokumentuje kolejny rozdział).

3.1. Wybór platformy systemowej

Na platformę systemową dla serwera gier wybrany został GNU/Linux — darmowy system operacyjny rozwijany przez wolontariuszy z całego świata. O wyborze przesądziło kilka czynników, z których najważniejsze to:

- GNU/Linux jest wolnym oprogramowanie (ang. *free software*, zob. [RWO]), dzięki czemu użytkownik ma dostęp do źródeł systemu, może je dowolnie modyfikować (dostosowywać do własnych potrzeb), używać i rozpowszechniać,
- GNU/Linux jako system uniksowy jest typowym systemem operacyjnym dla serwerów internetowych,
- GNU/Linux jako system serwerowy jest stosunkowo bezpieczny, stabilny i wydajny,
- GNU/Linux jest cały czas aktywnie rozwijany przez ludzi z całego świata.

3.2. Serwer gry

3.2.1. Wybór technologii

Jednym z wymagań niefunkcjonalnych serwisu była *wydajność* rozumiana jako możliwości obsługiwania jak największej liczby jednoczesnych użytkowników na danej platformie sprzętowej (fizycznym serwerze). Jako że głównym elementem serwisu z grami jest serwer gry (obsługuje on akcje wykonywane przez użytkowników podczas grania), wybór dla niego języka programowania i środowiska był kluczowy. Wybrany został język C++ z kilku powodów: programy napisane w C++ kompilowane są do kodu natywnego i wydajnością przewyższają języki interpretowane (Java, Python) dorównując programom pisanym w języku C; istnieje bardzo duży wybór bibliotek dla C/C++, dzięki czemu można korzystać z gotowych rozwiązań zamiast tworzyć wszystko samemu od zera; programy w C++ są jako tako przenośne na poziomie kodu źródłowego między różnymi uniksowymi systemami operacyjnymi; programując w C++ można praktycznie od razu wykorzystywać rozszerzenia jądra systemu (w niektórych

innych językach programowania trzeba czekać, aż zostaną wprowadzone odpowiednie zmiany do bibliotek języka).

3.2.2. Protokół komunikacji z elementami systemu

Na potrzeby komunikacji serwera gry z pozostałymi elementami systemu (tj. klientem gry, pośrednikiem, serwerem WWW) wykorzystywane są gniazda TCP/IP (komunikacja strumieniami, niezawodna). Ich wybór wynikał z wszechobecności w różnych środowiskach (językach), w których działają komunikujące się ze sobą elementy serwisu (programy w C++ i Javie po stronie serwera, aplet w Javie po stronie klienta, skrypt PHP), jak i prostoty.

Protokół komunikacji w strumieniach jest binarny, a dane są przesyłane paczkami; jedna paczka danych może zawierać dowolną liczbę danych liczbowych i tekstowych. Przewaga protokołu binarnego nad tekstowym jest tutaj dwojaka: znacznie ułatwione parsowanie danych i mniejsza ich ilość przesyłana między elementami systemu. Ilość przesyłanych danych jest szczególnie istotna przy komunikacji serwera gry z klientem i dlatego wszelkie dane są dodatkowo kompresowane, aby zminimalizować wymagania przepustowości łączy — zarówno tych, którymi podłączony jest serwer serwisu gier do Internetu, jak również łącza użytkownika końcowego. Do kompresji używany jest algorytm *Lempel-Ziv* (LZ77), którego implementację udostępnia biblioteka `zlib`.

3.2.3. Mechanizm badania deskryptorów

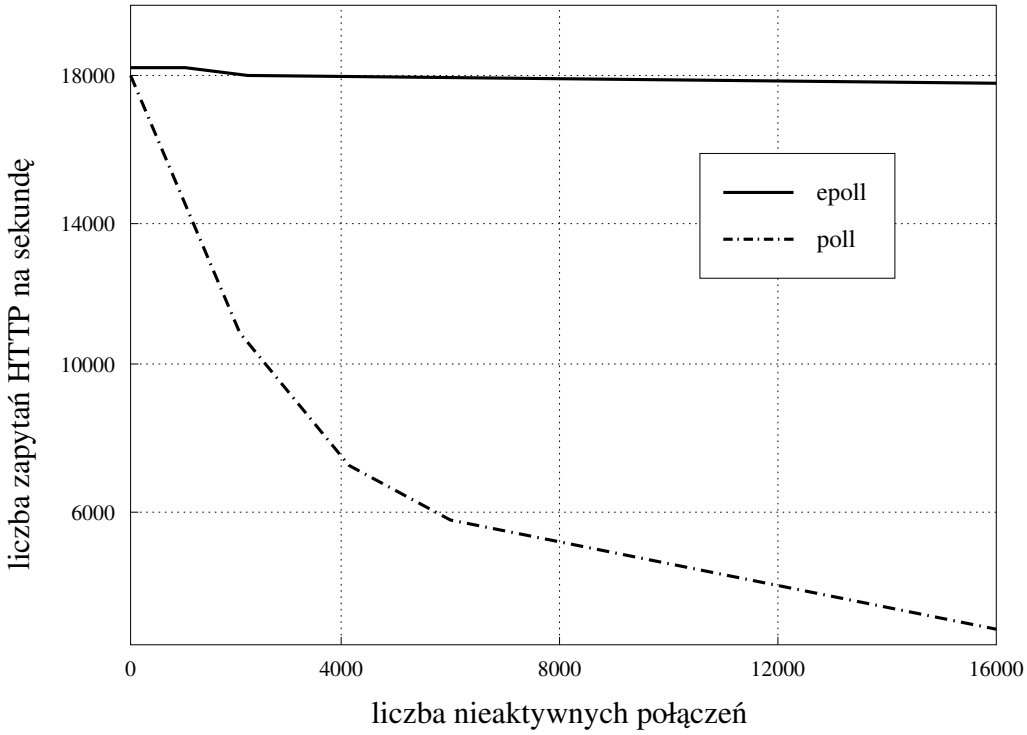
W uproszczeniu działanie serwera gry sprowadza się do odbierania danych od klientów gry (użytkowników), wykonywania dla nich określonych akcji i wysyłania ich wyników (zmian stanu) do klientów. Bezpośrednim zarządzaniem połączeniami z klientami zajmuje się stała liczba procesów pomocniczych serwera gry. Jeden taki proces obsługuje do tysiąca jednoczesnych połączeń i wynika to z ograniczenia liczby deskryptorów plików, jaką może jednocześnie używać proces. Do obsługi wielu połączeń (deskryptorów) przez jeden proces wykorzystuje się zwykle systemowe funkcje `poll` bądź `select`.

Obie one działają w podobny sposób. Jako parametr przyjmują zbiór deskryptorów, w którym dla każdego deskryptora określony jest warunek, czy oczekujemy na możliwość czytania z niego (`read`), czy pisania (`write`). Działanie funkcji kończy się albo po z góry określonym czasie, albo kiedy co najmniej jeden z deskryptorów będzie spełniał określony dla niego warunek. W wyniku otrzymujemy zbiór wszystkich deskryptorów spełniających ustalone dla siebie warunki.

Zarówno `poll` jak i `select` są implementowane przez jądro systemu operacyjnego w taki sposób, że co jakiś czas (ułamek sekundy) sprawdzane są **wszystkie** deskryptory przekazane jako parametr do funkcji. Takie sprawdzenie wszystkich deskryptorów musi występować co najmniej raz na wywołanie funkcji `poll` i `select` i wynika to z semantyki tych funkcji.

Niedawno zaproponowany został nowy mechanizm podobny w działaniu do funkcji `poll` i `select`, z tym że bada się w nim nie stan deskryptorów a zmiany stanu i nie wymaga on sprawdzania całego zbioru deskryptorów z każdym wywołaniem funkcji. Początkowo był on dostępny jako łata na jądro systemu z dostępem przez urządzenie `/dev/epoll`, a po pewnym czasie (październik 2002 r.) został włączony do rozwojowej wersji Linuksa 2.5.59 w postaci funkcji systemowej `sys_epoll` (zob. [EPL]).

Od strony programistycznej mechanizm dostępny jest przez zestaw kilku funkcji biblioteki `libepoll`, które różnią się semantycznie od `poll` i `select`. Na początku rejestrowany jest jeden deskryptor, z którym następnie kojarzy się zbiór deskryptorów do badania; ten deskryptor przekazuje się jako parametr dla funkcji `epoll_wait`; pomiędzy kolejnymi wywo-



Rysunek 3.1: Wydajności prostego serwera WWW w zależności od użycia funkcji poll lub epoll i liczby nieaktywnych połączeń

łaniami funkcji można dodawać deskryptory do skojarzonego zbioru i usuwać z niego. Funkcja `epoll_wait` przekazuje zbiór deskryptorów, których stan uległ zmianie od poprzedniego wywołania funkcji bądź od momentu dołączenia deskryptora do badanego zbioru; przez stan należy tutaj rozumieć możliwość zapisu/odczytu do/z deskryptora.

Wewnętrznie mechanizm jest zrealizowany w ten sposób, że dla zdarzeń takich jak pojawienie się danych do odczytu bądź miejsca do zapisu na danym deskryptorze `a` sprawdza się, czy należy on do zbioru deskryptorów skojarzonych z innym deskryptorem `b`. Jeśli tak, to w strukturze deskryptora `b` odnotowuje się ewentualną zmianę stanu deskryptora `a`. Działanie funkcji `epoll_wait` sprowadza się do sprawdzenia, czy w odpowiedniej strukturze jest odnotowana zmiana stanu dla jakiegoś deskryptora. Jeśli tak, to funkcja kończy swoje działanie, jeśli nie — zasypia i oczekuje na przebudzenie, które nastąpi w momencie zgłoszenia zmiany stanu dla któregoś z badanych deskryptorów.

Na rys. 3.1 przedstawiono wyniki testu prostego serwera WWW z użyciem mechanizmów `poll` i `epoll` (zob. [EPL]). Test badał wydajność serwera mierzoną w liczbie obsługiwanych zapytań na sekundę realizowanych przez 100 aktywnych połączeń przy różnej liczbie połączeń nieaktywnych. Widać na nim, że jeśli korzystamy z funkcji `poll`, to wydajność serwera WWW zależy istotnie od liczby nieaktywnych połączeń i maleje wykładniczo wraz z ich wzrostem, a gdy korzystamy z mechanizmu `epoll` wydajność jest praktycznie stała i nie zależy od liczby nieaktywnych połączeń.

Serwer gry wykorzystuje mechanizm `epoll`, jeśli jest on dostępny w systemie; jeśli nie jest, korzysta z funkcji `poll`, ale wydajność jego jest wtedy kilkukrotnie mniejsza.

3.3. Pośrednik

Pośrednik odpowiadający za komunikację z bazą danych został napisany jako wielowątkowy program w Javie. Fakt, że interpretowane programy w Javie są mniej wydajne od natywnych programów w C++ nie ma tutaj większego znaczenia, gdyż pośrednik wykonuje liczbę zadań nieporównywalnie mniejszą niż serwer gry (serwer gry obsługuje cały przebieg partii, podczas gdy pośrednik wykonuje tylko kilka operacji bazodanowych dopiero po jej zakończeniu).

Do komunikacji z bazą danych pośrednik wykorzystuje mechanizm JDBC (ang. *Java Database Connectivity*), którego podstawową zaletą jest jednolity sposób komunikacji niezależny od serwera bazy danych. Wymiana serwera bazy danych (np. z MySQL na Postgresql) wymaga jedynie wymiany sterownika JDBC.

3.4. Klient gry

3.4.1. Wybór technologii

Klient gry osadzany na stronie WWW został zaimplementowany jako aplet w Javie z kilku powodów: aplety w Javie są niezależne od architektury komputera, systemu operacyjnego czy przeglądarki, działają na dowolnej platformie, na którą istnieje maszyna wirtualna Javy; aplety w Javie są bezpieczne (przed uruchomieniem kod programu jest weryfikowany; sam nie może wykonywać żadnych operacji na lokalnym systemie plików, łączyć się z serwerami innymi niż ten, z którego pochodzi itd.); aplety w Javie są stosunkowo małe, głównie dzięki rozbudowanej standardowej bibliotece klas.

Alternatywą dla apletów w Javie jest technologia ActiveX, w której osadzony na stronie binarny program uruchamiany jest jak typowa lokalna aplikacja; ma pełny dostęp do lokalnego systemu plików, może w dowolny sposób korzystać z sieci itp. (innymi słowy nie jest bezpieczny, a użytkownik musi ufać, że program nie robi "nic złego"); co gorsze, technologia ta jest ograniczona do komputerów PC, systemu operacyjnego Windows i przeglądarki Internet Explorer.

W czasach, kiedy powstawała pierwsza wersja klienta gry, powszechna była wersja 1.1 środowiska Javy. Do konstrukcji interfejsu użytkownika ta wersja udostępniała bibliotekę AWT (ang. *Abstract Window Toolkit*); nie wszystkie wymagane komponenty wizualne (kontrolki) można było jednak zrealizować korzystając z klas tej biblioteki — niektóre musiały zostać zaimplementowane od zera (np. lista stołów, lista identyfikatorów graczy itp.).

Do komunikacji z serwerem gry klient wykorzystuje zwykłe gniazda TCP/IP; wybór innych technologii był w zasadzie niemożliwy ze względu na ich niedostępność w środowisku Javy 1.1 bądź brak wsparcia ze strony maszyn wirtualnych Javy rozpowszechnianych z przeglądarkami WWW, np. implementacja Javy w maszynie przeglądarki Internet Explorer 4.x nie miała wsparcia dla RMI (ang. *Remote Method Invocation*).

3.4.2. Klasy i ich podział

Na klienta gry składa się kilka pakietów zawierających klasy; te pakiety to:

- `components` — klasy definiujące wygląd i zachowanie ogólnych komponentów wizualnych (kontrolek), np. `ChatBoard` do wyświetlania wypowiedzi użytkowników, `NickList` do wyświetlania listy identyfikatorów),
- `components.a0` — podobnie jak `components`, zawiera klasy ogólnych komponentów wizualnych, ale wymaganych do odtwarzania rozegranych partii,

Rysunek 3.2: Wygląd klienta gry szachy

- `components.multi` — klasy wizualnych komponentów charakterystyczne dla gier wieloosobowych (`TableList` do wyświetlania listy stołów),
- `connect` — klasy do komunikacji z serwerem,
- `connect.jzlib` — klasy do dekompresji danych odbieranych od serwera,
- `games` — klasy określające wygląd niektórych okien i komponentów związanych z grami (`PreferencesFrame`, `UserInfoFrame`, `WhisperFrame`),
- `games.multi` — j.w., ale dla gier wieloosobowych,
- `games.multi.a0` — j.w., ale klasy wymagane tylko do odtwarzania rozegranych partii,
- `games.multi.board.backgammon` — klasy dla gry tryktrak: `Backgammon` (główna), `BackgammonBoard` (wygląd planszy do gry), `BackgammonTable` (zachowanie stołu do gry); dla innych gier planszowych (warcaby, reversi itp.) analogiczne klasy,
- `games.multi.card` — ogólne klasy dla gier karcianych (`CardBoard`, `CardTable`, `BiddingDialog`, `CardHistoryDisplay`, `LastTrickFrame` itp.)
- `games.multi.card.bridge` — klasy gry karcianej brydż: `Bridge`, `BridgeTable`, `BridgeBiddingDialog`; dla innych gier karcianych (skat, kierki itp.) analogiczne klasy,

Pliki z klasami skompilowane do kodu pośredniego (ang. *bytecode*) kompresuje się zwykle i umieszcza w archiwach `jar`. Ma to dwie podstawowe zalety: po pierwsze mniejszy rozmiar oznacza, że aplet ładuje się znacznie szybciej i mniejsze są wymagania łącza, którym serwis z grami podłączony jest do Internetu; po drugie pobieranie jednego pliku z wszystkimi klasami (zamiast każdej klasy w oddzielnym pliku) w mniejszym stopniu obciąża serwer WWW.

Przed kompresją kod klas poddawany jest też "zaczernieniu" (ang. *obfuscation*); usuwane są z niego zbędne informacje, nazwy zmiennych, metod i klas zmieniane są na jedno- bądź dwuliterowe. Zaczernianie ma na celu głównie utrudnienie dekompilacji programu i późniejszego zrozumienia, ale stosowane jest też wtedy, gdy chodzi jedynie o zmniejszenie rozmiaru klas.

W przypadku klienta gier rozmiar wszystkich klas wynosi 580KB; po skompresowaniu i umieszczeniu ich w archiwum `jar` rozmiar wynosi 345KB; jeśli przed skompresowaniem podda się kod klas zaczernieniu, to rozmiar archiwum z wszystkimi klasami wyniesie 317KB.

Pliki z klasami nie są jednak umieszczane w jednym archiwum `jar`, ale w kilku; nie miałoby to sensu, gdyby użytkownik chcąc zagrać w szachy pobierał klienty do wszystkich gier. Biorąc pod uwagę, że spora część klas jest wspólna dla wszystkich gier, a część dla pewnych grup gier, pakiety z klasami zostały rozmieszczone w archiwach w sposób następujący:

- `cm.jar` — klasy wspólne dla wszystkich gier, z pakietów:
  - `connect`,
  - `components`,
  - `components.multi`,
  - `games`,
  - `games.multi`;
- `co.jar` — klasy wspólne dla wszystkich gier, ale tylko te, które są wymagane do odtwarzania rozegranych partii; pakiety: `components.a0` i `games.multi.a0`;
- `cr.jar` — klasy wspólne dla gier karcianych; pakiet: `games.multi.card`;
- klasy specjalistyczne każdej z gier umieszczone w archiwum `XX.jar`, gdzie `XX` to dwuliterowy symbol gry (`ch` — szachy, `ck` — warcaby, `rv` — reversi itp.).

Przy takim podziale aplet do gry np. w szachy wymaga archiwów `cm.jar`, `co.jar` i `ch.jar`, a aplet tylko do odtwarzania rozegranych już partii w szachy wymaga jedynie `co.jar` i `ch.jar`. Po pobraniu archiwów wymaganych do gry w szachy przejście na grę warcaby wymaga jedynie pobrania niewielkiego archiwum `ck.jar` (klasy charakterystyczne dla gry warcaby), gdyż pozostałe są wspólne dla obu gier.

3.4.3. Wielojęzyczność

Klient gry został zaimplementowany w taki sposób, aby możliwa była obsługa wielu języków, w jakich pokazywane są napisy i komunikaty tekstowe, bez konieczności zmiany samego klienta. Zostało to zrealizowane w ten sposób, że klient nie ma w sobie zaszytych żadnych tekstów, ale otrzymuje je od serwera zaraz po nawiązaniu z nim połączenia; jako pierwszy wysyła informacje, w jakim języku chce napisy i późniejsze komunikaty.

Kilka napisów, które pojawiają się jeszcze przed nawiązaniem kontaktu z serwerem (np. "proszę czekać..", "próba połączenia z serwerem zakończona porażką.."), są przekazywane do klienta gry na stronie WWW jako parametry.

3.5. Serwis WWW

3.5.1. Wybór technologii

Jako serwer WWW, którego zadaniem jest udostępnianie stron WWW, wybrany został Apache HTTP Server. Apache to obecnie najpopularniejszy serwer WWW używany przez ponad 50% serwisów internetowych, z publicznie dostępnym kodem źródłowym (zob. [APC]).

Do generowania stron o charakterze dynamicznym (np. z aktualnymi statystykami graczy, listą rankingową itp.) wybrany został mechanizm PHP. PHP jest prostym językiem skryptowym do generowania stron WWW, wspierającym wiele standardów internetowych (HTTP, FTP, ciasteczka itp.), baz danych (MySQL, PostgreSQL, Oracle itp.) i bibliotek programistycznych (bzip2, mcrypt, zlib). Podobnie jak serwer Apache, PHP jest udostępniane nieodpłatnie z kodami źródłowymi (zob. [PHP]).

W przypadku serwisów aktywnie wykorzystujących mechanizm PHP niezbędne jest używanie różnego rodzaju akceleratorów, których działanie polega na przechowywania w pamięci podręcznej skompilowanych wersji skryptów PHP (ang. *caching*). Dzięki temu przy każdorazowym odwołaniu do stron wykorzystujących PHP pomijana jest faza parsowania i kompilacji kodu, co istotnie poprawia wydajność serwera WWW — według niektórych testów zauważalny jest nawet **kilkukrotny** wzrost wydajności. Do akceleracji PHP w serwisie gier wybrany został PHP Accelerator, wykorzystywany m.in. przez firmę Yahoo! i udostępniany nieodpłatnie, ale... bez kodu źródłowego (zob. [PHA]).

Stosunkowo mało powszechną, ale bardzo pożyteczną, jest praktyka kompresji stron WWW podawanych przez serwer WWW. Wsparcie dla kompresji zostało wprowadzone w wersji 1.1 protokołu HTTP i jest obecne w praktycznie wszystkich przeglądarkach internetowych, w tym nawet tak starych jak Internet Explorer 3.0. Do kompresji stron wybrany został `mod_gzip` — moduł do serwera Apache, który (jak nazwa wskazuje) wykorzystuje kompresję gzip (zob. [MGZ]).

3.5.2. Strony WWW serwisu

Wszystkie dynamicznie generowane strony serwisu WWW korzystają ze wspólnych funkcji i danych umieszczonych w jednym pliku o nazwie `ut.php`. Zawiera on:

- funkcje bazodanowe (nawiązywanie połączenia z bazą, wykonywanie zapytań itp.),
- funkcje do komunikacji z serwerem gry (źródłem danych prezentowanych na stronach może być również serwer gry; zapytania do serwera gry dotyczą głównie liczby obecnych użytkowników, obecności określonego użytkownika itp.),
- funkcje i dane związane z wyglądem stron (podstawowe kolory używane na stronach, funkcje generujące nagłówki i stopki stron), dzięki czemu wszystkie strony serwisu wyglądają podobnie,
- dane o adresach stron (zmiana adresu jakiejś strony, np. do logowania się, wymaga jedynie zmiany w tym pliku, gdyż wszystkie strony z niego korzystają; adresy nie są zaszyte na sztywno w treści stron),
- dane o dostępnych grach (nazwa, katalog z główną stroną gry, port serwera gry, nazwy plików z archiwum klienta gry i wymaganych archiwów pomocniczych, nazwa głównej klasy klienta gry itp.)

Rysunek 3.3: Wygląd strony głównej

Sam wygląd stron jest dosyć ascetyczny; na stronach nie ma zbędnej grafiki, dzięki czemu czas ich ładowania jest bardzo krótki. Dodatkowo dzięki kompresji stron (z wykorzystaniem modułu `mod_gzip`) ich rozmiar z kilkunastu kilobajtów jest zmniejszany do średnio 2-3KB, co sprawia, że strony ładują się błyskawicznie, nawet gdy użytkownik korzysta z modemu.

Obsługa sesji na stronach została zrealizowana bez korzystania z mechanizmu ciasteczek (ang. *cookies*). Kiedy użytkownik się loguje, przydzielany jest mu 32-znakowy identyfikator sesji, który pomiędzy stronami przekazywany jest jako parametr (np. `/szachy.php?session=-12c8hf8zk5vzxoj23hfsdf12oas7dn12`) Takie rozwiązanie jest rzadko spotykane, ale nie wymaga, aby przeglądarka internetowa akceptowała ciasteczka (użytkownik może wyłączyć ich obsługę np. ze względu na ochronę prywatności).

3.6. Baza danych

3.6.1. Wybór technologii

Praktycznie wszelkie dane związane z serwisem gromadzone są w bazie danych, która oparta jest o serwer MySQL. Jednym z głównych powodów wyboru właśnie tego serwera baz danych była jego wydajność, którą zawdzięcza po części ograniczonej funkcjonalności (brak jest na przykład wsparcia dla kluczy obcych, podzapytań itp.). Serwer MySQL jest udostępniany z

kodami źródłowymi na licencji GPL.

3.6.2. Optymalizacje

Niektóre z zapytań wykonywane na bazie danych wymagały poważnych zabiegów, aby czas ich realizacji był rozsądny. Na przykład jedna ze stron ze statystykami gracza przedstawia listę jego przeciwników z bilansem gier wygranych, przegranych itp. Zapytanie, które generuje tę listę, w przypadku popularnych gier wymaga złożenia ze sobą tabeli `participants` zawierającej ponad dwa miliony rekordów z samą sobą i tabelą `plays` zawierającą milion rekordów.

Optymalizacje i zabiegi, które pozwoliły skrócić czas wykonywania zapytań, to:

- ograniczony rozmiar danych — np. przez użycie typu `tinyint` zamiast `int`; dzięki temu, że dane są niewielkie, większa ich ilość może znajdować się w cache'u systemu operacyjnego; to z kolei sprawia, że czas dostępu do nich jest znacznie krótszy niżby to miało miejsce, gdyby dane musiały być odczytywane z dysku;
- stały rozmiar rekordów — ułatwia wyszukiwanie danych na dysku (wystarczy pomnożyć numer rekordu przez wielkość) i zmniejsza fragmentację danych; szczególnie istotne w przypadku tabel, które są często aktualizowane;
- podział dużych tabel na kilka mniejszych
- grupowanie zapytań — operacje dotyczące aktualizacji tabeli przechowującej dane związane z zarządzaniem sesjami użytkowników są grupowane i wykonywane jako jedno zapytanie zamiast kilku prostych, rozłożonych w czasie;
- ograniczony czas przechowywania danych — informacje o grach rozegranych dawniej niż pół roku są automatycznie usuwane;
- wykonywanie raz na jakiś czas defragmentacji danych w plikach bazy danych (komenda `OPTIMIZE TABLE ...`), sortowanie indeksów itp.
- odpowiednia konfiguracja serwera bazy danych, m.in. przez dobór ilości pamięci przeznaczonej na bufor dla indeksów (parametr `key_buffer`), bufor do sortowania (parametr `sort_buffer`) itp.

Rozdział 4

Wdrożenie

W niniejszym rozdziale przedstawiono historię i statystyki serwisu internetowego o nazwie Kurnik, który został uruchomiony w czerwcu 2001 r. (zob. [KUR]).

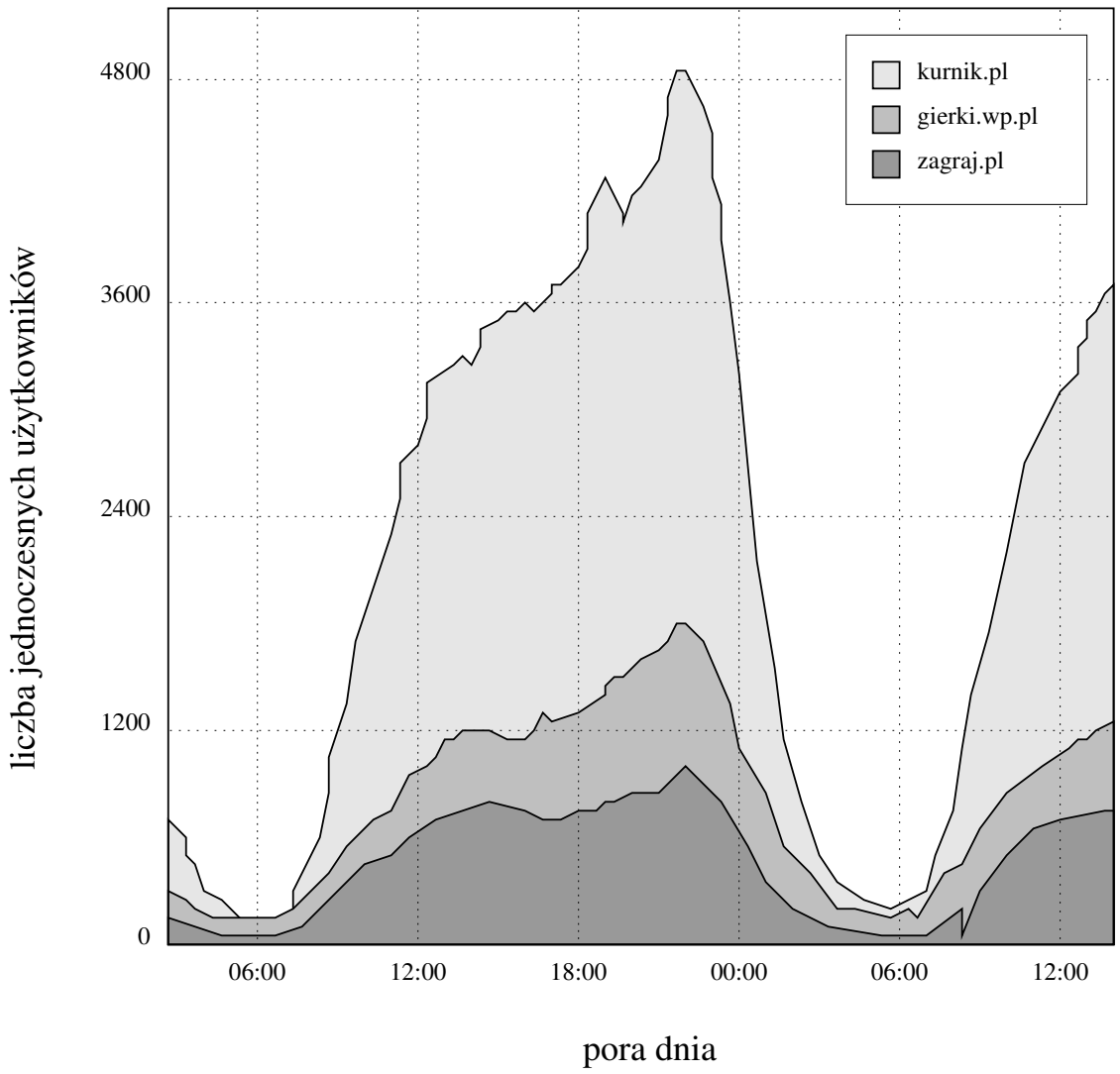
4.1. Historia

Pierwsza wersja serwisu uruchomiona w czerwcu 2001 r. zawierała 5 gier planszowych: szachy, warcaby, gomoku, reversi i szkrable. W grach nie było dostępnych żadnych statystyk, rankingów itp. Kiedy serwis pojawił się w sieci, na polskim rynku istniały już dwa inne serwisy z grami: jeden prowadzony przez firmę Cronix Sp. z o.o. (zob. [CRO]), a drugi będący częścią portalu Wirtualna Polska (zob. [WPG]).

W sierpniu 2001 r. w serwisie pojawiły się gry karciane (3-5-8, brydż, kierki, pan, planowanie, tysiąc), a niecałe dwa miesiące później statystyki graczy i system mierzenia ich siły w postaci drabinki.

W roku 2002 kolejne miesiące przynosiły coraz to nowe gry, ulepszenia, zmiany itp. W marcu pojawiły się cztery nowe gry karciane (kierki amerykańskie, makao, piki, skat), miesiąc później dwie gry planszowe (młynek, tryktrak) i ranking wzorowany na rankingu szachowym ELO, a w maju jeszcze dwie karciane (remik gin, ogórek). W tym samym miesiącu z serwisu musiała też zostać usunięta gra szkrable (ze względów prawnych), a w jej miejsce pojawiała się podobna gra o nazwie literaki. Jako że z gier korzystało coraz więcej cudzoziemców nieznających języka polskiego, w sierpniu 2002 r. uruchomiona została angielska wersja językowa serwisu. W tym samym miesiącu doszła też jedna nowa gra planszowa (czwórki). Dosyć interesujące ulepszenie w postaci przechowywania przebiegów partii w grach planszowych pojawiło się w październiku 2002 r.

Pierwsza połowa roku 2003 przyniosła nieco mniej nowości niż analogiczny okres roku poprzedniego. W lutym pojawiły się dwie nowe gry planszowe (go, hex) oraz angielska wersja gry literaki. W marcu zorganizowany został konkurs na logo serwisu, w którym wzięło udział ponad 100 osób, a w maju ogłoszona została zbiórka pieniędzy na zakup nowego serwera na potrzeby serwisu. Reakcja użytkowników na apel była fenomenalna — w niespełna dwa tygodnie udało się zebrać ponad 20 tys. zł. W tym miejscu wypada dodać, że Kurnik działa cały czas jako serwis o charakterze niekomercyjnym (korzystanie z niego jest bezpłatne, na stronach nie ma żadnych banerów reklamowych itp.), podczas gdy dwa pozostałe polskie serwisy z grami są przedsięwzięciami komercyjnymi, które pod koniec 2002 roku wprowadziły nawet opłaty za korzystanie z gier.



Rysunek 4.1: Rozkład liczby jednoczesnych użytkowników w ciągu dnia

4.2. Statystyki

Popularność serwisu Kurnik rosła i nadal rośnie z każdym miesiącem. Obecnie (maj 2003) liczba użytkowników korzystających z serwisu w ciągu miesiąca wynosi ponad 160 tys., a liczba zarejestrowanych kont to blisko 500 tys.

Z technicznego punktu widzenia interesujący wydaje się fakt, że serwis działa wciąż na "zwykłym składaku" z procesorem AthlonXP 1700MHz i 1.5GB pamięci RAM (zakup nowego serwera jest w trakcie realizacji), a mimo to w godzinach szczytu (zob. rys. 4.1) potrafi obsłużyć prawie 5 tys. **jednoczesnych** użytkowników. Zużycie procesora dochodzi wtedy do blisko 80%, a wskaźnik *load* na serwerze waha się w granicach 3-10.

Godna uwagi jest też zajętość łącza, którym serwer jest podłączony do Internetu. W godzinach szczytu, przy prawie 5 tys. jednoczesnych użytkowników, wynosi ona około 4Mbps (megabity na sekundę); gdy jednak nie jest włączona kompresja protokołu HTTP i danych przesyłanych od serwera gier do klientów, to zajętość wynosi prawie dwukrotnie więcej, około 7Mbps.

Statystyki serwera WWW przedstawiają się następująco: wielkość pliku z logiem odwołań

do serwera w jednym dniu przekracza 400MB; liczba odwołań do serwera (ang. *hits*) wynosi ponad 2 mln. dziennie, z czego 700 tys. to odwołania do stron (ang. *page views*), a pozostałe do plików graficznych, archiwów JAR itp.

Najwięcej odwołań ze wszystkich stron posiada strona ze statystykami (176 tys.), następnie strona z apletem gry (139 tys.), strona główna (103 tys.) i strona logowania (102 tys.). Główne strony najpopularniejszych gier pobierane są w liczbach: 39 tys. (gra tysiąc), 27 tys. (literaki), 15 tys. (3-5-8), 12 tys. (szachy), 10 tys. (brydż).

Inne statystyki: liczba różnych adresów IP, z których następują odwołania do serwera, przekracza 24 tys.; w godzinach szczytu serwer WWW obsługuje około 40 zapytań na sekundę, a serwer bazodanowy — ponad 120.

Rozdział 5

Podsumowanie

Opisany w pracy serwis gier działa w Internecie od czerwca 2001 r. pod adresem `www.kurnik.pl`. Można nieskromnie powiedzieć, że stanowi on udaną realizację serwisu z grami, o czym może świadczyć chociażby stale rosnąca liczba jego użytkowników (w maju 2003 wynosiła ponad 160 tys.). Sama liczba użytkowników może nie jest jeszcze żadnym wyznacznikiem, ale ich lojalność wyrażona przez ofiarne wsparcie finansowe na zakup nowego serwera ma duże znaczenie.

Akceptacja ze strony użytkowników była możliwa dzięki temu, że serwis spełnia w dużej mierze najważniejsze wymagania funkcjonalne przedstawione w pierwszym rozdziale tej pracy. O prostocie i wygodzie korzystania z serwisu może świadczyć to, że na jego stronach brak jest jakiejkolwiek instrukcji obsługi, a intuicyjność interakcji i interfejsu użytkownika pozwalała korzystać z serwisu nawet cudzoziemcom nieznającym języka polskiego (kiedy jeszcze nie było angielskiej wersji językowej).

Warto też podkreślić udaną realizację serwisu od strony technologicznej. Architektura serwisu i wybór technologii, w których zostały zrealizowane poszczególne jego elementy, zapewnia w miarę wysoką wydajność, o czym świadczy dosyć duża liczba użytkowników (około 5 tys.), jaką może obsługiwać jednocześnie pojedynczy serwer niskiej klasy.

Istnieje kilka sposobów, na które serwis mógłby być ulepszony — najlepszy z nich to dodanie automatycznej organizacji turniejów. Uzupełnienia wymaga też zestaw dostępnych gier, brakuje bowiem jeszcze kilku klasycznych, takich jak domino, kanasta czy mankala.

Spis literatury

[GTG] *The Online Guide to Traditional Games*, http://www.tradgames.org.uk/

[ELO] *Elo rating system*, http://www.wikipedia.org/wiki/Elo_rating_system

[RNK] *Ranking.PL – Poznaj trendy w Internecie*, http://www.ranking.pl/

[RWO] *Ruch na rzecz Wolnego Oprogramowania*, http://www.rwo.org.pl/

[EPL] */dev/epoll Home Page*, http://www.xmailserver.org/linux-patches/nio-improve.html

[APC] *The Apache HTTP Server Project*, http://httpd.apache.org/

[PHP] *PHP: Hypertext Preprocessor*, http://www.php.net/

[PHA] *The ionCube PHP Accelerator*, http://www.php-accelerator.co.uk/

[MGZ] *mod_gzip Information Page*, http://i4net.tv/marticle/mod_gzip/

[MYS] *MySQL: The World's Most Popular Open Source Database*, http://www.mysql.com/

[JVA] *The Source for Java Technology*, http://java.sun.com/

[ACX] *ActiveX Controls*, http://www.microsoft.com/com/tech/ActiveX.asp

[JZL] *JZlib – zlib in pure java*, http://www.jcraft.com/jzlib/

[GPL] *GNU General Public License*, http://www.gnu.org/licenses/gpl.html

[OSS] *Open Source Initiative OSI*, http://www.opensource.org/

[KUR] *Kurnik – gry online*, http://www.kurnik.pl/

[CRO] *Zagraj.pl (dawniej Cronix)*, http://www.zagraj.pl/

[WPG] *Wirtualna Polska – gry online*, http://gierki.wp.pl/